Community Detection (CD)

Loose definition: identifying "dense" subgraphs in some network

We consider communities in various different ways:

- Friend groups
- Groups with a shared interest
- Basic interactions

Recall our Karate network:

- Zachary Karate club
- Club split in two
  - ↳ between club president and the club instructor
- Interesting aspect: the split followed almost perfectly along a topological boundry in the network

... a topological boundry in the network

→ Fundamental hypothesis of community detection

AKA communities can be defined or described fully via network topology

What we'll be discussing:

- How to define "density"
- Algorithms used for CD
- How to evaluate CD algo. outputs
- Related problems (graph partitioning)

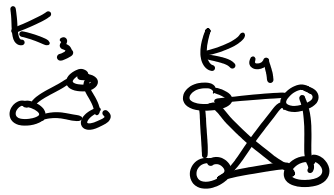
Density Definitions for CD

"Density" ~ lots of edges within a given subgraph

How dense can we get? K_1 K_2 K_3 K_4

↳ cliques

↳ cliques



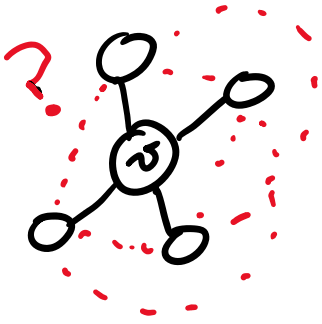
Can we define communities as cliques?

Issue: large cliques are rare in real networks

Issue: computationally difficult
* NP-complete in general
* Equivalent to subgraph matching
* $O(1.19^n)$ bound for cliques

However: triangles can be very useful (K_3) 
* density ~ clustering coefficient
* complexity isn't bad $O(m^{3/2})$

Issue: extrapolating local clustering to a broader region is tough

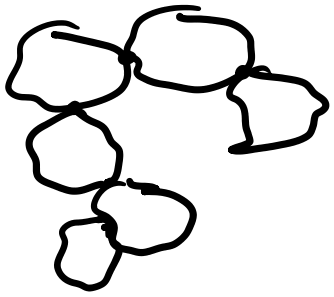


↳ communities can have a large diameter

Can we use connectivity?

* Connected components → not useful
* k-connected components → more useful

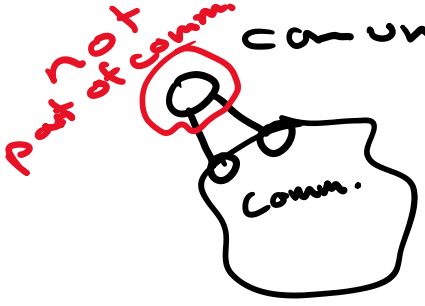
maximal k-connected subgraphs → still connected after removing k vertices



* Captures same notion of "cut"

Issue: tough to compute O(polytime)

Issue: connectivity doesn't directly correspond to notions of communities



K-cores: same but different

Better definition: relative edge density

→ ratio of internal to external edges of a subgraph

strong community: $\forall v \in C : d_{int}(v) \geq d_{ext}(v)$

weak community: $\sum_{v \in C} d_{int}(v) \geq \sum_{v \in C} d_{ext}(v)$

→ Both give explicit measures of density

AND: a global "quality" measure

Issue: trivial optimal solution is just one community with all vertices

Other better measures:

Modularity and conductance

Number of Communities

Number of communities is unknown, will vary based on optimization criteria and network topology

Challenges:

* We have no ground truth
* Possible per-vertex groupings given some groups → exponential

given some groups → exponential

* Possible groups → super-exponential

↳ solution space is "quite large"

Takeaway: Heuristics and greedy algorithms are used in practice

---

CD Algorithms

Types:

* Agglomerative: we combine communities to reach some local maximum for some optimization

* Divisive: we cut communities in some way to reach local maximum for optmization

* Hierarchical: we create a hierarchical structure of communities through some approach

↳ Often the case in practice

Example

Note: communities can also overlap (very difficult)

Label Propagation

Agglomerative, can be hierachical

Iterative:

For all $v \in V(G)$:

$C(v)$ = max community that shows up in $N(v)$

Pros: simple to implement

$\sim O(n)$ complexity

good in practice

good in practice

Cons: Can have bad results
optimal solution is single comm.
hierarchy is tough to infer

Ravasz Algorithm

Agglomerative

Iterate until a single community
Select some (C_i, C_j) community to merge based on a similarity metric (common neighbors)

Pros: full hierarchy captured
can modify our similarity metric

Cons: complexity not great $O(n^2)$
no global metric being optimized

Girvan-Newman

Divisive

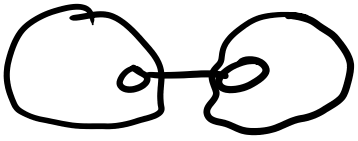
Iterate
select edge e with the

Iterate

select edge e with the highest betweenness centrality and cut it

↳ communities are connected comps

Note: we're cutting weak ties or local bridges



Pros: works well, as it inherently utilizes a lot of basic network properties

Cons: slowwwwwwwwwww $O(n^3)$